Tech Support Article Titles

A tech support article consists of a title topic and one or more page topics. The title and the page topics all have a common name except for the last two digits. For example the first article two pages. The topics are:

0635Article0100 -- This is the topic that is displayed on the monitor as the mouse rolls over the post it.

0635Article0101 -- this is page one

0635Article0102 -- this is page two

If you want to add more pages, make those topics 0635Aritcle0103, etc. If you want only one page, delete topic 0635Article0102.

Jumps are done by:

Select the text you want to be the jump.

Format it as double underline.

Immediately after the "hot" text, insert <gototopic> where gototopic is the context string (footnote in the .rtf file) for the topic you want to go to. If you want to go to something other than page one of the topic, append a :2 where '2' is replaced with the page number you want to go to.

Format <gototopic> as hidden text (no underline).

Getting jumps to actually work is a little tricky. MediaView tends to get confused easily. If you have several jumps near each other and some of them don't work try:

1) Add more space between the lines. Make the font bigger. Both will help.

2) Make absolutely sure there is not text that gets cut off the bottom. If there is extra text insert a page break and make sure all the text fits. See Kent for more details.

Colors:

For tech support articles text formatted as "auto" color (Format...Font menu in MS Word for Windows) appears as white. Text formatted with other colors appear that color (red, green, blue, cyan,...)

Number of articles:

You can add as many of articles as you want. If there are not enough postits just put them in the index.

WARNING: MediaView is a little brain dead at times. If you use the indent paragraphs button on the MS-Word tool bar, hot spots will stop working (sometimes). See Kent for full details.

WARNING: Do NOT use paragraph break in topics with jumps. They will not work. Instead us soft break (shift ENTER). See Kent for all the details.

Technical Support
Article Index

Index

Index (cont'd)

DOS for Gamers

Memory Management Made Easy

by Tim Victor

DOS for Gamers: Memory Management Made Easy
by Tim Victor

DOS for Gamers was featured in the January ‘95 PC Gamer

DOS memory management can be confusing and frustrating, or it can be an enjoyable intellectual challenge. It all depends on your state of mind. Don't think of configuring your PC's memory as a chore, think of it as a type of computer game. There are puzzles to solve, trade-offs to be made, an arcane language to decode (hexadecimal arithmetic), and a rich payoff for succeeding - the knowledge that your computer will have enough free memory to run any program you might encounter.

Like so many things in life, confidence is the key - and confidence comes from knowing enough about the problem to believe that you can solve it. You have to understand what memory is. You have to be familiar with **EMM386.EXE**, the driver that creates upper memory blocks (**UMBs**), and with the **DEVICEHIGH** and **LOADHIGH** commands which load drivers and resident programs into these regions of usable memory outside the normal **DOS** area. But once you have them down, mastering **DOS** memory management is really just a matter of practice and experience.

OK, But What is Memory? I could literally write one hundred pages on the subject, but that idea was vetoed. So here's the quick-and-dirty version.

First off, memory is not hard-drive space. Hard drives are for storage; think of them as filing cabinets. Memory, or **RAM**, is used by your computer to temporarily keep files and programming code open, so that your computer doesn't have to search for it on the hard drive and use it a little bit at a time. So memory is like a table top, where you can spread out the contents of a file (which you get from that hard-drive filing cabinet) and work on them all at once. The more memory you have, the bigger your table top and the more files you can spread out and work on at once. If you don't have enough memory, it'd be like reading a 100-page report by going to the file cabinet, taking one page, laying it on your table and reading it, then walking back to the cabinet, refilling that page and taking the next one... you get the idea.

Of course, it's not quite that easy; there are different kinds of memory, too. But explaining all that would take forever, so once again, here's a condensed explanation. The most important thing to remember is that your computer doesn't look at all of its available **RAM** in the same way. The memory up to **640K** is called conventional, or base, memory - this is the area programs are referring to when they say -not enough memory,' meaning that you don't have enough free base **RAM** to load the program. The memory between **640K** and **1MB** is upper memory, and this is where you should be able to stuff all those device drivers and Terminate and Stay Resident (**TSR**) programs that use up your base memory. And then there's the rest of your memory, the stuff above **1MB** - and it can be configured

as Expanded memory (**EMS**) or Extended memory (**XMS**), or a little of each. If you aren't loading an expanded memory manager like **DOS**'s **EMM386.EXE** in your **CONFIG.SYS**, you'll have **XMS** by default.

What follows, then, is intended to help you free up that all-important base memory, so you can get your games up and running. Memory management is pretty technical stuff, so keep your **DOS** manual handy to check out any terms you might not be familiar with.

You Gotta Start Somewhere . Take a look at my PC's setup:

[AUTOEXEC.BAT]

C:\ADDSTOR\SSTOR D: /M=C:\ADDSTOR\SUPERSTR.DRV

C:\DOS6\MSCDEX.EXE /D:MSCD001 /M:10

C:\DOS6\SMARTDRV.EXE

PATH C:\DOS6;D:\BIN;C:\WIN31;C:\QQEMM;C:\ADDSTOR;C:\NU

SET TEMP=C:\WIN31\TEMP

SET TMP=C:\WIN31\TEMP

SET BLASTER=A220 I5 D3 H5 T4

[CONFIG.SYS]

DOS=HIGH,UMB

DEVICE=C:\DOS6\HIMEM.SYS

DEVICE=C:\DOS6\EMM386.EXE NOEMS HIGHSCAN I=B000-B7FF

DEVICE=C:\ADDSTOR\SSTORDRV.SYS /MAXMOUNT=1 /NOAUTO

DEVICE=C:\CDROM\MTMCDE.SYS /D:MSCD001 /P:300 /A:0 /M:64 /T:S /I:10

DEVICE=WIN31\MOUSE.SYS

DEVICE=C:\REELMAGC\RMDRV.SYS

Like anyone's, it needs a collection of drivers and **TSRs** to run. Right now, I'm not loading any of those **TSRs** high (above the 640K limit), but I am telling **EMM386** - **DOS'** memory manager - to be very aggressive about creating upper-memory blocks (UMBs). UMBs are the segments of upper memory where I'll eventually load those **TSRs. EMM386's NOEMS** keyword creates **UMBs** but turns off support for **EMS** memory emulation (we'll get to that in a minute) and the **HIGHSCAN** keyword replaces unused areas of **ROM** with **UMB's**, while the **I=B000-B7FF** option manually creates a **27K** upper memory block in a region normally reserved for the monochrome graphics adapter.

If you scan through the startup files (**CONFIG.SYS** and **AUTOEXEC.BAT**), you'll see that it's mostly standard stuff except for a few unusual items. **SSTORDRV.SYS** is the driver for SuperStor, the disk compression software that I use, while **SSTOR** is a simple program that actually installs the compressed disk as a **D:** drive. **RMDEV.SYS** is the driver for my Sigma Designs ReelMagic board. (I don't have that many ReelMagic games, but since I also use the board as my sound card, the driver has to stay.) Otherwise it's pretty typical: a mouse driver, a **CD-ROM** driver and **MSCDEX,** the Microsoft **CD-ROM** extensions, and the **MS-DOS's** SmartDrive disk cache utility.

One thing you'll use often as you configure your system's memory is the **MEM** utility included with **DOS**. Just type **MEM** from any prompt (followed by /C for a more detailed report, and /F for a list of free **UMBs**) to get a tally of your system's memory usage. Right now, the **MEM** utility (see chart below) shows that I only have **517K** of free base memory to load programs in, and I'd like to have a lot more than that: Few games these days will run if you have less than **560K** free, and many want a lot more than that. SmartDrive is the only program that automatically loads itself into upper memory, so there's plenty of room for improvement. Since there are **162K** bytes free in the largest **UMB** and less than **120K** of conventional memory is being used, this should be a cinch.

[Chart 1 Using MEM/C Command]

Modules using memory below 1 MB:

Name	Total		=	Conventional		+	Upper Memory	
MSDOS	13,053	(13K)		13,053	(13K)		0	(0K)
HIMEM	1,168	(1K)		1,168	(1K)		0	(0K)
EMM386	3,120	(3K)		3,120	(3K)		0	(0K)
SSTORDRV	29,328	(29K)		29,328	(29K)		0	(0K)
MTMCDE	20,800	(20K)		20,800	(20K)		0	(0K)
MOUSE	17,088	(17K)		17,088	(17K)		0	(0K)
RMDEV	2,352	(2K)		2,352	(2K)		0	(0K)
COMMAND	2,928	(3K)		2,928	(3K)		0	(0K)
MSCDEX	36,224	(35K)		36,224	(35K)		0	(0K)
MSDOS	18,208	(18K)		0	(0K)		18,208	(18K)
SMARTDRV	30,368	(30K)		0	(0K)		30,368	(30K)
Free	704,640	(688K)		529,168	(517K)		175,472	(171K)

Memory Summary:

Type of Memory	Total	=	Used	+	Free
Conventional	655,360		126,192		529,168
Upper	224,048		48,576		175,472
Reserved	131,072		131,072		0
Extended (XMS)	7,378,128		2,454,736		4,923,392
Total memory	8,388,608		2,760,576		5,628,032
Total under 1 MB	879,408		174,768		704,640

Largest executable program size	528,976	(517K)
Largest fee upper memory block	166,224	(162K)

MS-DOS is resident in the high memory area.

[Chart 2 Using MEM/F Command]

Free Conventional Memory:

Segment	Total	
015D8	80	(0K)
015EE	96	(0K)
01ECC	208	(0K)
01ED9	88,992	(87K)
03493	440,000	(430K)
Total Free:	529,376	(517K)

Free Upper Memory

Region	Largest Free		Total Free		Total Size	
1	9,200	(9K)	9,248	(9K)	27,456	(27K)
2	166,224	(162K)	166,224	(162K)	196,592	(192K)

It's Almost Too Easy. With **192K** available in the second **UMB**, you could park a bus up there. In this configuration, loading everything high is just a matter of changing -**DEVICE='** lines to -**DEVICEHIGH='** in **CONFIG.SYS**, and inserting -**LH'** (short for -**LOADHIGH**') at the start of certain lines in **AUTOEXEC.BAT**. The -/**L:2'** option is a manual instruction telling **DOS** to use the second UMB region. We don't even have to bother with that piddly **27KB UMB**. And with **620K** bytes free for **DOS** programs, it's hard to complain about the outcome.

Now It Gets Harder. Success stories are nice, but that last example was a little bit artificial. For instance, there are still some games that need **EMS** memory emulation to access memory beyond one megabyte, and that in turn requires a **64K** -'**EMS** page frame' located in the upper memory area. I used **MS-DOS's** MemMaker program to automatically configure my system with **EMS** enabled, and here's what I got (see **MSD** Chart below). Besides scrambling up my nice, neat startup files, it left me with **575KB** available. Some people could probably get by with that, but I've seen too many programs that needed **600K** or more to run.

As the memory map in **DOS's** Microsoft Diagnostic utility (type **MSD** from the **DOS** prompt) shows, the biggest problem is where MemMaker put the EMS page frame, because it now splits one large **UMB** into two smaller ones. There's enough free space up there for SuperStor's driver, a **30K** improvement, but the space has to be contiguous. Right now, it's split into two separate segments.

[Chart Using MSD]

----------Memory----------

Legend: Available " " RAM "##" ROM "RR" Possibly Available ".."

EMS	Page	Frame "PP"	Used	UMBs "UU"	Free UMBs "FF"
1024K	FC00	RRRRRRRRRRRRRRRR		FFFF	Conventional Memory
	F800	RRRRRRRRRRRRRRRR		FBFF	Total: 640K
	F400	UUUUFFFFFFFFFFFFFF		F7FF	Available: 574K
960K	F000	UUUUUUUUUUUUUUUU		F3FF	588464 bytes
	EC00	PPPPPPPPPPPPPPPP		EFFF	
	E800	PPPPPPPPPPPPPPPP		EBFF	Extended Memory
	E400	PPPPPPPPPPPPPPPP		E7FF	Total: 7424K
896K	E000	PPPPPPPPPPPPPPPP		E3FF	
	DC00	FFFFFFFFFFFFFFFFFF		DFFF	MS-DOS Upper Memory Blocks
	D800	UUUFFFFFFFFFFFFFF		D8FF	Total UMBs: 154K
	D400	UUUUUUUUUUUUUUUU		D7FF	Total Free UMBs: 49K
832K	D000	UUUUUUUUUUUUUUUU		D3FF	Largest Free Block: 14K
	CC00	UUUUUUUUUUUUUUUU		CFFF	
	C800	UUUUUUUUUUUUUUUU		CBFF	Expanded Memory (EMS)
	C400	RRRRRRRRRRRRRRRR		C7FF	Version: 3.01
768	C000	RRRRRRRRRRRRRRRR		C3FF	Page Fram Address: E000H
	BC00	##################		BFFF	Total: 7744K
	B800	##################		BBFF	Available: 4848K
	B400	UUUUUUFFFFFFFFFFFF		B7FF	
704K	B000	#####UUUUUUUUUUU		B3FF	XMS Information
	AC00			AFFF	XMS Version: 3.00
	A800			ABFF	Driver Version: 3.10
	A400			A7FF	A20 Address Line: Enable
640K	A000			A3FF	High Memory Area: In use
					Available: 4608K
					Largest Free Block: 4608K
					Available SXMS: 4608K
					Largest Free SXMS: 4608K
					VPCI Information
					VCPI Detected: Yes
					Version: 1.00
					Available Memory: 4848K

A Definite Improvement. With EMM386's -FRAME=' option relocating the EMS page frame to the C800 segment, we get a solid 128KB **UMB** to work with - but it still takes a couple more tricks to get above the 600K mark. **MTMCDE**, my CD-ROM driver, fits nicely in region 1 (see **MEM** chart below), the 27K **UMB** in the monochrome graphics area, but I had to load **MOUSE.SYS** in conventional memory to fit everything else. Still, that only cost me 17K, which isn't too bad.

[MEM Chart]

Modules using memory below 1 MB:

Name	Total		=	Conventional		+	Upper Memory	
MSDOS	13,053	(13K)		13,053	(13K)		0	(0K)
HIMEM	1,168	(1K)		1,168	(1K)		0	(0K)
EMM386	3,120	(3K)		3,120	(3K)		0	(0K)

MOUSE	17,088	(17K)	17,088	(17K)	0	(0K)
COMMAND	2,928	(3K)	2,928	(3K)	0	(0K)
SSTORDRV	47,520	(29K)	0	(0K)	47,520	(46K)
MTMCDE	20,832	(20K)	0	(0K)	20,832	(20K)
RMDEV	2,384	(2K)	0	(0K)	2,384	(2K)
MSCDEX	36,224	(35K)	0	(0K)	36,224	(35K)
SMARTDRV	30,368	(30K)	0	(0K)	30,368	(30K)
Free	704,640	(688K)	617,872	(603K)	29,504	(29K)

Memory Summary:

Type of Memory	Total	=	Used	+	Free
Conventional	655,360		37,488		617,872
Upper	158,560		129,056		29,504
Reserved	131,072		131,072		0
Extended (XMS)	7,378,064		2,725,024		4,718,592
Total memory	8,388,608		3,022,640		5,365,968
Total under 1 MB	879,920		166,544		647,376

Total Expanded (EMS)	7,929,856	(7,744K)
Free Expanded (EMS)	4,964,352	(4,484K)

* EMM386 is using XMS memory to simulate EMS memory as needed.
Free EMS memory may change as free XMS memory changes.

Largest executable program size	617,776	(603K)
Largest fee upper memory block	22,768	(22K)

MS-DOS is resident in the high memory area.

Before SmartDrive, the last program on the list, would load in upper memory, I also had to shrink the memory requirements of **MSCDEX** by cutting it from 10 to 6 buffers (the -/M:6' option). I'm using SmartDrive 5.0, part of **MS-DOS** versions 6.20 and later, which will cache a **CD-ROM** drive as long as it loads after **MSCDEX**, so I'm not too concerned about a loss of speed from using fewer buffers. But like any other change, it's best to test to make sure it isn't a problem.

If you looked at the output from the **MEM** command, you might have noticed that the largest free upper memory block contains **22KB**, which should have been enough to hold the **17K** mouse driver. But when I tried to load it high, SmartDrive dropped back into conventional memory, netting me a **13KB** loss. The problem is that most **DOS** drivers and **TSRs** take up more space when they start themselves up than when they're loaded and resident. In each **UMB** region, the last module that loads will leave some slack space, the difference between its size when it loaded and its final size. It's kind of frustrating to see that space sitting idle, and third-party memory managers like Quarterdeck's **QEMM** and Qualitas' **386M**ax offer ways to temporarily stretch **UMBs** while a module loads. But this is the best we can do with **EMM386**.

I use an accelerated video card, which makes Windows run much faster, but there's a big catch: when it's running in high display resolutions, the video card needs the entire address space from **A000** to **C7FF.** Without the **27K UMB** that we found down there, available memory would drop back to about **580KB. EMM386** has done fairly well so far,

but we've already pushed it to its limit, so my standard setup uses Quarterdeck's **QEMM** memory manager. I even use the dreaded -Stealth mode', which locates **UMBs** on top of video and **BIOS ROMS**, then remaps the **ROMs** through the **EMS** page frame as they're needed.

The **MSD** memory map shows how it works: the single **UMB** starts at **C200**, where video **ROM** would normally be and, with most of the **BIOS ROM** mapped out, the **EMS** page frame sits way up at the **EC00** segment. Everything loads in upper memory, and the **QEMM.SYS** driver module also replaces **MS-DOS's HIMEM.SYS,** even though it's smaller than **EMM386.SYS** alone. That nets another thousand or so bytes, for a total of over **621K** available. Stealth mode has picked up a reputation for instability and incompatibility, and maybe it is on some systems. As the saying goes, your mileage may vary, but I wouldn't want to try to get along without it.

Knowing When You've Done Enough. There's a nerdy kind of satisfaction that comes with getting every device driver and **TSR** loaded into upper memory. If you get to that point, you can count on having about **620K** of conventional memory available, enough to run any game out there. But if it simply isn't possible to get everything loaded high, that isn't necessarily cause for despair; only a few games are so greedy that they demand much over **600KB**, and with the increasing popularity of **DOS** extenders, that first **640K** of memory is becoming less and less precious. For example, Doom will run with less than **400K** of conventional **DOS** memory free, as long as you have a couple megabytes of

extended memory available for it to use. In general, **600KB** of free **DOS** memory will be do the job most of the time, and that's an achievable goal for most systems.

Playing It Safe Maximizing **DOS** memory usually means a lot of testing and experimenting. More than half the time, a change will actually make things worse, and there's always the chance that the system will lock up when you reboot to test a change. There are a couple steps that you can take to reduce the risk, though.

First of all, make safety copies of your **AUTOEXEC.BAT** and **CONFIG.SYS** files before you start working, so that if things go badly, you can still back up to where you started. And when you make a change to a line, leave the original line in the file and make your changes to a copy of the line. Just put the word **-REM'** (for -remark') at the start of the line so that **DOS** knows to ignore it.

As of **MS-DOS** version **6.00**, if the system hangs while it's starting up, you can almost always get it to boot by holding down the Shift key or **F5** before **DOS** loads. This tells **DOS** to ignore the **CONFIG.SYS** and **AUTOEXEC.BAT** files entirely and to start with a totally clean system. You can also press the **F8** key to step through **CONFIG.SYS** one line at a time, which helps a lot when you're trying to figure out which line is making the system hang. With these emergency exits available under **DOS 6**, boot floppies aren't needed as often as they used to be, but it's still a good idea to have one around. If you need to create one, just type **-FORMAT A: /S'** at the **DOS** prompt, then put in a new disk or one that you don't mind erasing - formatting will wipe out all the data on the disk. And that's all there is to it!

Take Out the Trash:

A **PC's CONFIG.SYS** and **AUTOEXEC.BAT** files are sort of its basement and attic; we only go in there when we absolutely have to, and we can't remember where half of the junk in there came from in the first place. Stuff tends to collect in those files - lines added by programs that you once installed, a driver for a card that you don't use anymore, or things that you don't understand but they've always been in there. If one of these lines loads a device driver or program that you don't really need, it wastes valuable memory.

So before you burn hours trying to get everything in both files to load high, go through each file line-by-line and ask yourself whether that line really needs to be there. Look it up in the manual if you have to. You'll find lines lurking inside your **CONFIG.SYS** that load drivers like **ANSI.SYS**, which you might be able to live without, and **EGA.SYS**, which you almost certainly don't need, and lines that set unnecessarily large values for the **FILES**, **BUFFERS**, and **STACKS** keywords. Some programs require special settings for these but, most of the time, **DOS's** default values are fine. In **AUTOEXEC.BAT**, look for **TSR** programs like **SETVER.EXE**, **DOSKEY.COM** and **SHARE.EXE**, which you might not need, and for other utilities that you can live without. If you need them, fine - but if you don't, get rid of 'em.

Sound Problems

Where’s the sound?

Sound Problems
Where's the sound?

If you are not receiving any sound, the problem is generally caused from having the wrong sound card selected or the game not programmed to recognizing the sound card you have.

Often when installing a demo, you are prompted to setup your sound card. At times this is detected automatically, but at others you must manually select the type of card and often include the ***IRQ, DMA and Addresses*** values.

Since these are demos, all sound issues may not be fully worked out. Try selecting alternate or compatible options, and you may want to double check your ***IRQ*** and other addresses (these can often be found in your ***AUTOEXEC.BAT*** file).

See tips on ***Sound Card Addresses***

Game Lock-Ups

When a game

leaves you hanging

Game Lock-Ups: When a game leaves you hanging

There are three main reasons for games to lock up when you go to play them. These relate to ***Sound Card***, ***Memory*** or ***Video*** issues.

FIRST: Check the sound card setup (see the "***Sound Problems***" Postit). This is the most common cause of lock-ups for demos and is the quickest and easiest to troubleshoot. It's a good place to start.

SECOND: The next most common reason is a lack of memory. It can be a lack of ***Conventional*** ***Expanded*** or ***Extended*** memory. Make sure you've read the game's system requirements to ensure you have enough total **RAM**. If you do, see the separate Postits on each of these three types of memory.

THIRD: The third problem may involve Video and Graphic compatibility. Some games may not detect your specific video card. If the game's install procedure has video options, recheck these, if they seem correct you may want to try loading a ***UNIVESA*** driver if you have one.

If it's a Windows game (version 3.1), you may need to check your video driver and resolution under the Windows Setup icon (usually in your Main folder).

Windows 3.1 Games that Appear Distorted

Windows 3.1 Games that Appear Distorted

If you find some **Windows 3.1** games appear distorted, there is a Windows **QTW.INI** file you can edit to try and correct this. The **QTW.INI** file should be located in your **\WINDOWS** subdirectory. Call this game up in your **DOS** editor and locate the section heading **[VIDEO**]. See if this section contains the line **OPTIMIZE=DRIVERS**. If the line reads **OPTIMIZE=HARDWARE**, change it. If the line doesn't exist, add it.

Using Expanded Memory

Using Expanded Memory

Some games require that you boot your PC with as much Expanded memory (**EMS**) as possible. To do this you may need to alter your **CONFIG.SYS** file *or* create a boot disk with an altered **CONFIG.SYS** file.

The key here is to make sure the **NOEMS** statement *DOS NOT* reside on the **EMM386** or **QEMM386** memory management line in your **CONFIG.SYS** file. You'll need to load the ***CONFIG.SYS*** file into a **DOS** editor. Locate the **EMM386** or **QEMM386** device line, it should look something like this:

device=C:\DOS\EMM386.EXE

or

device=C:\DOS\EMM386.EXE NOEMS

You'll want to remove the **NOEMS** extension and change the line to read something like:

device=C:\DOS\EMM386.EXE RAM

It may also have some other extensions at the end of this line, you can normally leave those in. Just to be safe, instead of rewriting the device line, you can enter the word **REM** in front of your original line and add the **RAM** line separately. It would look something like this:

REM device=C:\DOS\EMM386.EXE NOEMS
device=C:\DOS\EMM386.EXE RAM

The **REM** statement will prevent that line from loading. This way, if you need to switch back, you can reload your **CONFIG.SYS** file and remove the **REM** statement and add it in front of the **RAM** statement line instead.

Sound Card Addresses
(IRQ HELL)

Sound Card Addresses (IRQ HELL)

Upon installing some games you will be prompted to enter your sound card's **Address**, **DMA** and **IRQ** locations. This is not to be confused with **MIDI** addresses.

The most common configuration is: **Address=220**, **DMA=1** and **IRQ=5**, though **IRQ=7** is also very common.

If you are not sure what these are for your sound card, there is often a line in your **AUTOEXEC.BAT** file that will identify these. The line will look something like this:
SET BLASTER=A220 I7 D1 H5 P330 T6

In this example, the **A220** refers to your **Address** being **220**. The **I7** refers to the **IRQ** location of **7**, and **D1** refers to the **DMA** location of **1**. You'll also notice **P330**, this is usually, but not always, your **MIDI** address.

Once you know these details, you can run the **SETUP** for the demo again and make the necessary changes. If the game has no separate sound setup, you may need to reinstall the game.

OPERATING THE PC GAMER
DOS SHOOT

OPERATING THE PC GAMER DOS SHOOT

The **PC Gamer DOS Shoot** was designed to help you install **DOS** games from Windows.

When you're ready to install a **DOS** game from the ***Jukebox***, you're getting ready to ride the **DOS Shoot**. When you first click on the Install button for the game, you'll receive a message. ***Read this message carefully!*** This message will tell you exactly what to type, because you're going to actually shut down Windows and end up at the **DOS** prompt.

When you are finished installing, or even playing the demo, you can type **WIN** at any **DOS** prompt and you'll reenter Windows and end up at the **Jukebox**, just where you left off. Yes, it's possible, as long as you haven't rebooted your PC in the interim.

Conventional Memory Management

Conventional Memory Management

Conventional memory, and issues pertaining to it, are best explained in the ***DOS for Gamers*** article. Just click on ***DOS for Gamers*** and we'll hyperlink you there.

Extended Memory Management

Extended Memory Management

Some games require that you boot your PC with as much ***Extended memory*** as possible. To do this you may need to alter your **CONFIG.SYS** file or create a boot disk with an altered **CONFIG.SYS** file.

This is the opposite of trying to free up ***Expanded Memory***. The key here is to add the ***NOEMS*** statement to your **EMM386** or **QEMM386** memory management file. You'll need to load the **CONFIG.SYS** file into a **DOS** editor. Locate the **EMM386** or **QEMM386** device line, it should look something like this:

device=C:\DOS\EMM386.EXE

or

device=C:\DOS\EMM386.EXE RAM

What you'll want to do is change the line to read: **device=C:\DOS\EMM386.EXE NOEMS**

It may also include some other extensions at the end of this line such as **HIGHSCAN** (**device=C:\DOS\EMM386.EXE RAM HIGHSCAN**). You will want to remove this. Just to be safe, instead of rewriting the device line, you can enter the word **REM** in front of your original line and add the **NOEMS** line separately. It would look something like this:

REM device=C:\DOS\EMM386.EXE RAM HIGHSCAN
device=C:\DOS\EMM386.EXE NOEMS

The **REM** statement will prevent that line from loading. This way, if you need to switch back, you can reload your **CONFIG.SYS** file and remove the **REM** statement and add it in front of the ***NOEMS*** statement line.

Running DOS Games with Windows 3.1

Running DOS Games with Windows 3.1

You should **NEVER** load a **DOS** game through **Windows 3.1** (installing, configuring or playing). This can cause havoc to your Windows environment. If the game is designated **DOS**, don't attempt to load through Windows, don't even use the **DOS SHELL** for this purpose. You need to close Windows by using the **ALT-F4** keys or clicking on the box in the top left corner.

We make mention of this because our Windows based front-end carries **DOS** game demos.

The exception to this rule is the ***DOS Shoot*** that you can enter through the **Jukebox**. The DOS Shoot actually closes Windows, so there will be no conflicts.

Running DOS Games with Windows ‘95

Running DOS Games with Windows ‘95

Our **PC Gamer demo CD** contains games designed for three separate operating systems; **DOS**, **Windows 3.1** and **Windows ‘95**. At the ***Jukebox***, the games are divided into these three categories.

If you are using **Windows ‘95** as your operating system, the **Windows ‘95** and **Windows 3.1** games should install and run directly from our CD. The **DOS** games may force you to enter a **DOS mode** and install and run the games from **DOS**.

Occasionally, when a **DOS** game can install or run under **Windows ‘95**, it will also be included under the **Windows ‘95 Games** button. So, if it’s a **DOS** game you’re interested in, check under the **Windows ‘95** button first.

The Jukebox

The Jukebox

A major feature of our **PC Gamer CD** is the **Jukebox**. The **Jukebox** contains all the game demos, patches, add-ons and any other interesting program we come across.

The **Jukebox** is divided into six categories, represented by the six buttons at the bottom of the **Jukebox** screen. These categories are; **DOS Games**, **Windows 3.1 Games**, **Windows '95 Games**, **Bug Patches**, **Add-ons**, and **Online Networks**.

Click on a button and you'll be presented with a list of all the programs for that category. If there are more programs than what will appear on the monitor at one time, you can click on the arrow-down button on the right (when it's lit up) and more programs will scroll up.

Once you've entered a category, you can click on the individual program you're interested in. At this point you'll be presented with options to either install and run the program or notes on how you can use the program from our **CD**.

One of the features of our **Jukebox** is a ***DOS Shoot***, which will let you close down your **Windows** application and install or run a game from **DOS**. And when you're done, you can reenter the **PC Gamer CD** right where you left off.

TIPS ON USING THE PC GAMER CD

TIPS ON USING THE PC GAMER CD

Here are a few tips on how to get around our **PC Gamer** office:

1.If you want to get to the demos as quickly as possible, you can skip the intro and elevator ride. To skip the intro, click on the bottom of the screen when the logo first appears. To skip the elevator ride, click on the jukebox instead of the door handle. This will take you directly to the ***Jukebox*** where all the programs are located.

2.To **exit** the game, move your mouse to the top of the screen. A pull down menu should appear with an **EXIT** option. This also contains options to turn off sound effects. Or, you can return to the elevator to **exit**.

3.There are basically six sites you can visit inside the office. These are the **Receptionist Desk**, **Editors Desk**, **Technical Support Desk**, **Jukebox**, **Reviews Desk**, and **File Cabinet**. Just move your mouse to the left or right side of the screen and when it turns into large arrows you'll have the option to move in that direction.

4.At the **Receptionist Desk**, you can click on the phone to receive messages or ring the bell to get the receptionists attention. At times she won't answer, usually when you've

repeatedly called her. There's also a magazine rack to receive order information on **various Imagine Publications**. For fun, you can click on *Cheryl's* coffee mug.

5.At the **Editor's Desk**, there are two sub sites you can enter. To the left is a phone system where you can click on the redial button to see who *Dan* last called. You can also dial up a phone number, which will occasionally be linked to the internal game. To the right is a letter *Dan* is currently working on. There's no telling what this might say. It too might, at times, be linked to the game. For fun, click on the hand-drawn picture on the wall.

6.The **Technical Support** desk has two subsites. One revolves around the monitor with sticky notes attached. If you scan your mouse over these notes, messages will appear on the monitor, but you already know that or you wouldn't be reading this. The messages normally contain hyper-linking so you can access related information. The other site is a **Rolodex**. This contains addresses and technical support numbers for most gaming companies.

7.The ***Jukebox*** is where all the programs are kept.

8.The **Reviews Desk** contains the **PC Gamer Top Ten List**.

9.The **File Cabinet** is a virtual cornucopia of information. The top drawer is an archive of the **Table of Contents** for every issue - very handy if you're interested in back issues. The second drawer is our **Reviews Index**. It contains a list of every game we've ever reviewed and includes a brief description, ratings and which issue they appeared in. The bottom drawer has a combination and is used for portions of the **PC Gamer** game. You've gotta do the right thing to get the combination.

10. There is a game associated with using the **PC Gamer** CD. Clues are left around the office. You'll know when you've located all of the clues when you press the bell at the **Receptionist Desk** and receive a voice-over from our **PC Gamer** Narrator.

Playing it Safe

Playing it Safe

Maximizing **DOS** memory usually means a lot of testing and experimenting. More than half the time, a change will actually make things worse, and there's always the chance that the system will lock up when you reboot to test a change. There are a couple steps that you can take to reduce the risk, though.

First of all, make safety copies of your **AUTOEXEC.BAT** and **CONFIG.SYS**. files before you start working, so that if things go badly, you can still back up to where you started. And when you make a change to a line, leave the original line in the file and make your changes to a copy of the line. Just put the word -**REM**' (for -remark') at the start of the line so that **DOS** knows to ignore it.

As of **MS-DOS** version 6.00, if the system hangs while it's starting up, you can almost always get it to boot by holding down the **Shift** key or **F5** before **DOS** loads. This tells **DOS** to ignore the **CONFIG.SYS** and **AUTOEXEC.BAT** files entirely and to start with a totally clean system. You can also press the F8 key to step through **CONFIG.SYS** one line at a time, which helps a lot when you're trying to figure out which line is making the system hang. With these emergency exits available under **DOS 6**, boot floppies aren't needed as often as they used to be, but it's still a good idea to have one around.

If you need to create one, just type -**FORMAT A: /S'** at the **DOS** prompt, then put in a new disk or one that you don't mind erasing - formatting will wipe out all the data on the disk. And that's all there is to it!

Cleaning Up Your AUTOEXEC & CONFIG Files

Cleaning Up Your AUTOEXEC & CONFIG Files

A **PC's CONFIG.SYS** and **AUTOEXEC.BAT** files are sort of its basement and attic; we only go in there when we absolutely have to, and we can't remember where half of the junk in there came from in the first place. Stuff tends to collect in those files - lines added by programs that you once installed, a driver for a card that you don't use anymore, or things that you don't understand but they've always been in there. If one of these lines loads a device driver or program that you don't really need, it wastes valuable memory.

So before you burn hours trying to get everything in both files to load high, go through each file line-by-line and ask yourself whether that line really needs to be there. Look it up in the manual if you have to. You'll find lines lurking inside your **CONFIG.SYS** that load drivers like **ANSI.SYS**, which you might be able to live without, and **EGA.SYS**, which you almost certainly don't need, and lines that set unnecessarily large values for the **FILES**, **BUFFERS**, and **STACKS** keywords. Some programs require special settings for these but, most of the time, **DOS's** default values are fine. In **AUTOEXEC.BAT**, look for **TSR** programs like **SETVER.EXE**, **DOSKEY.COM** and **SHARE.EXE**, which you might not need, and for other utilities that you can live without. If you need them, fine - but if you don't, get rid of 'em.

Loading VESA Drivers

Loading VESA Drivers

Some games won't load when it cannot detect the video card you use. When this occurs, you often need to load a separate **VESA** driver. **VESA** stands for **Video Electronic Standards** Association. You will normally load this **VESA** driver from **DOS**. The **VESA** driver you load depends on your video card manufacturer. If a driver was not included with the purchase of your machine or video card, you may need to contact the manufacturer to see which one will work with your card. These **VESA** drivers can normally be found on most major online services.

If you have a **UNIVESA** driver, try loading it. Otherwise, you may want to call the game's technical support number and see if your video card is supported by that game.

Don’t forget to call Red’s

Don’t forget to call Red’s.

Catfish

Catfish Fillets

Cornmeal

Flour

Oil

Paprika

Pepper

Okra

Spices

Combine two parts flour with one part cornmeal in bowl. Wash catfish fillets, leave wet. Sprinkle paprika, pepper and other preferred spices (garlic, hot peppers, cajun-type seasoning - whatever suites your taste). Dip fillets into flour/cornmeal mixture (no need to dip in eggs or other batter). Heat up oil in iron skillet on medium high. Catfish fillets don't need to fry too long, just until the outside gets lightly brown. Since you've got cornmeal, spices and a hot pan of oil out, it's the perfect time to fry up some okra using the same steps as the catfish.

The PC Gamer Game

The PC Gamer Game

There is a game associated with using the **PC Gamer** CD. Clues are left around the office. These clues will direct you toward text files or audio files. You'll have to do a bit of snooping around to uncover everything.

You'll know when you've located all of the clues when you press the bell at the **Receptionist Desk** and receive a voice-over from our **PC Gamer** Narrator. Read the **Disc Pages** in the magazine for additional information on our little game, and what to do when you've uncovered all the clues.

Notes on Add-on Levels

Notes on Add-on Levels

We have add-on levels (often referred to as WADS) for the following games: **DOOM2**, **HERETIC**, **HEXEN**, **DESCENT**, **WARCRAFT**, and **THE PERFECT GENERAL 2**.

When you click on the add-on level from the jukebox you're interested in, you'll receive information on the exact filename and where the file is located on our CD. The file will need to be copied to your Hard Drive where the full version of the game is located.

The **DOOM2** levels are located in the **\DOOM** directory.
The **HERETIC** levels are located in the **\HERETIC** directory
The **HEXEN** levels are located in the **\HEXEN** directory
The **DESCENT** levels are located in the **\DESCENT** directory
The **PERFECT GENERAL 2** level is located in the **\PATCHES** directory

NOTE: These levels are not separate games in themselves. They are additional levels to be used with the full versions of the games (if you don't have those full versions, you can't use these levels). **Also**, if the file has a **ZIP** extension. It needs to be unzipped using **PKUNZIP** (not provided with this CD).

There is a **README.TXT** file on how to use these levels provided in each of the directories on our CD. If you're not familiar with how to install these levels once you've copied and unzipped them, here are some tips. (click the down arrow button to the right).

For the **DOOM2**, **HERETIC**, and **HEXEN** levels, each filename will have a **WAD** extension (which is why they're called **WADS**). Once it's copied into the game's directory on your Hard Drive, you can load them by typing the executable command followed by **-FILE** and the filename for the **WAD**. For example, if the game is **DOOM2** and the filename is **EVILDOG.WAD**, you'd type:
DOOM2 -FILE EVILDOG.WAD.

For **DESCENT** levels it's even easier. Just copy the file into the game's directory on your Hard Drive. Load **DESCENT** and the new level should appear as a new mission for you to select.

For **WARCRAFT**, copy the files into the game's directory on your Hard Drive. You'll need to rename the file as a saved game. You can rename it as **SAVEGAME1.SAV**, **SAVEGAME2.SAV** all the way up to **SAVEGAME7.SAV**. The level will appear in your save games. **NOTE**: This will overwrite any saved game you had in that slot.

For **PERFECT GENERAL 2**, copy the files into the game's scenario directory on your Hard Drive. The game should appear in your list of campaigns.

Notes on Using Bug Patches

Notes on Using Bug Patches

Patches are not games. They are files that correct problems or add enhancements to games you already have.

All patches are located in the **\PATCHES** subdirectory on the CD. If you are interested in using one of these patches, you'll need to exit our **PC Gamer** frontend and copy the patch to the directory on your Hard Drive where that game resides.

Click on the name of the game for the patch you are interested in using from the Jukebox. A message will appear giving you the exact filename of the patch and a description of what the patch does.

Further information about these patches can be found in the magazine under the **Extended Play** column by **Tom McDonald**.

For example, if the name of the patch is **APACHE1.ZIP** and your **CD-ROM** drive is **D:**. You would go to your **D** drive and type **CD\PATCHES**. Then copy the **APACHE1.ZIP** file to the directory on your Hard Drive where the Apache program resides.

When a patch has the **ZIP** extension, it will need to be unzipped using **PKUNZIP** (not provided with this CD).